Food Tastes Best When It's On Time

FOOD DELIVERY OPTIMIZATION

Presented by Group 1

MUHAMMAD AKMAL BIN WANAHARI (215666)
MUHAMMAD AQIL IQBAL BIN MOHD JAMIL (215728)
MUHAMMAD FAIZ BIN MD FADZLI (215768)
ABDUL MAJID BIN MISRUJI (214727)
MUHAMMAD HAZLAM AZWAR BIN MOHD FARIZAL (215838)

Team Members

MUHAMMAD AKMAL BIN WANAHARI
215666

MUHAMMAD AQIL IQBAL BIN MOHD JAMIL
215728

MUHAMMAD FAIZ BIN MD FADZLI
215768

ABDUL MAJID BIN MISRUJI
214727

MUHAMMAD HAZLAM AZWAR BIN
MOHD FARIZAL
215838

Problem Statement

- The city faces a surge in online food orders.
- Common complaints include late deliveries, cold food, and missed time windows.
- Delivery assignments are currently manual and routes are inefficient, with overlapping and backtracking trips.

Goal

- Develop an automated delivery assignment and routing system that minimizes total delivery time and distance, while satisfying operational constraints.

Why is it important?

- ⏱️ Faster deliveries = happier and more loyal customers.
- 🍱 Food quality is preserved when delivery is prompt.
- 💰 Minimizing travel time reduces fuel and operational costs.
- 📈 Riders can complete more deliveries per shift, improving productivity.
- 🚦 The system can adapt to real-time orders and changing traffic conditions, increasing robustness.

Problem Specification

Inputs:

A set of food orders with:

- Delivery locations (coordinates)
- Order time
- Delivery time window (acceptable delivery range)

Rider fleet with:

- Current location
- Maximum order capacity

Traffic data (optional for advanced optimization)

Restaurant location (starting point)

Outputs:

- An optimized delivery route for each rider that:
  - Minimizes total distance
  - Satisfies delivery time windows
  - Balances rider workload

Constraints:

- Each rider can only carry a limited number of orders.
- Delivery must happen within promised time windows.
- Each order can only be delivered once.
- Riders start from the restaurant location.

Objectives

Minimize delivery time & distance.

Handle multiple orders per rider.

Respect delivery time constraints.

Be scalable for real-time operations.

Expected Output

Optimized Delivery Plan

Total delivery cost printed for both GA and SBA

Unit Testing Results

Time Execution for both GA and SBA

Algorithm Overview

Algorithm Design Techniques

- Algorithm design helps solve hard real-world problems efficiently.
- Includes classic methods (like DAC) and smart techniques (like GA).
- Uses these methods to create better solutions for computing tasks.

Importance of Algorithm

- Choosing the right algorithm helps solve problems faster and better.
- Different problems need different types of algorithms.
- For example, greedy is good for quick choices, DP is best for complex steps.

Algorithm Designs

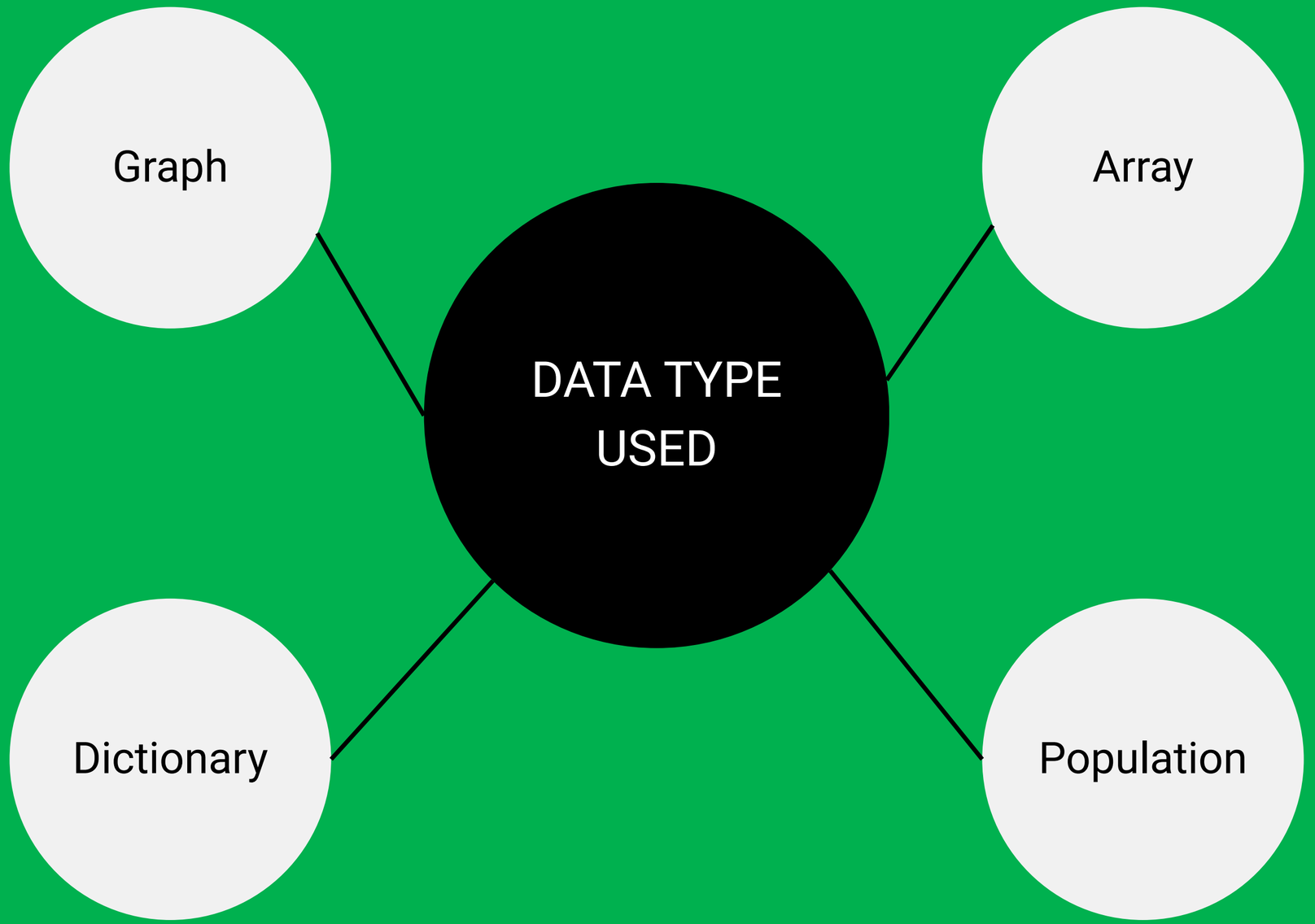
1. **Greedy Algorithm**
   Picks the best, fast and easy
2. **Dynamic Programming (DP)**
   Remember, more memory, step by step
3. **Divide and Conquer (DAC)**
   Split problem, solve & join, searching
4. **Graph Algorithm**
   Nodes, short paths, used in maps
5. **Sorting Algorithm**
   In order, various, search & organize

Comparison of algorithms

- To find which one works fastest and uses less resources.
- To see which algorithm fits the problem best.
- To understand the pros and cons of each method.

ALGORITHM	GREEDY	DP	DAC	GRAPH + GA	SORTING
STRENGTH	Fast & Simple	Hard problems	Big problems	Flexible	Organizing
WEAKNESS	Contraints	Memory	Dependant	Time to set up	Delivery route
TIME COMPLEXITY	O(n log n)	$O(n^2)$ to $O(n^3)$	O(n log n)	Varies	O(n log n)
SUITABILITY	Not Suitable	Not Suitable	Not Suitable	Best Choice	Partially

ALGORITHM PARADIGM


Graph
Array
DATA TYPE USED
Dictionary
Population

Objective Function

Minimizing Total Delivery Cost

- Faster deliveries enhance customer satisfaction and optimize rider schedules
- Efficient routes save fuel, cutting operational costs
- Smart path selection avoids congestion, reducing delays and fuel waste

Penalty-Based Constraint Handling

- Promotes routes that meet all constraints, ensuring deliveries are both possible and practical
- Helps avoid late deliveries and overworked riders by guiding the algorithm toward balanced solutions

Constraint

Space

Riders have limited order capacity, influencing route planning and feasibility

Time

Deliveries must meet specific time windows and adapt to real-time traffic conditions

Value

Limits on rider distance, order count, and route validity (no duplicates or missing locations)

Pseudocode Graph + Genetic Algorithm

```
Procedure GeneticAlgorithmForFastBite()

    Input:
        Graph G(V, E)          // V: delivery points, E: routes with
distance/time/cost
        Orders[]               // Each order has location, time window
        Riders[]               // Each rider has capacity limit
        PopulationSize         // Number of candidate solutions
        MaxGenerations         // Total generations to evolve
        CrossoverRate          // Probability of crossover
        MutationRate           // Probability of mutation

    Output:
        BestRouteAssignment    // Optimized delivery routes for
all riders

    Begin
        Population ← GenerateInitialPopulation(G, Orders,
Riders, PopulationSize)
        EvaluateFitness(Population)

        For generation ← 1 to MaxGenerations Do
            NewPopulation ← empty set

            While size(NewPopulation) < PopulationSize Do
                Parent1 ← Select(Population)
                Parent2 ← Select(Population)

                If Random() < CrossoverRate Then
                    Offspring1, Offspring2 ← Crossover(Parent1, Parent2)
                Else
                    Offspring1 ← Parent1
                    Offspring2 ← Parent2
                EndIf

                If Random() < MutationRate Then
                    Offspring1 ← Mutate(Offspring1)
                EndIf

                If Random() < MutationRate Then
                    Offspring2 ← Mutate(Offspring2)
                EndIf

                EvaluateFitness(Offspring1)
                EvaluateFitness(Offspring2)

                Add Offspring1 to NewPopulation
                Add Offspring2 to NewPopulation
            EndWhile

            Population ← SelectNextGeneration(Population,
            NewPopulation)
        EndFor

        BestRouteAssignment ← GetBestSolution(Population)
        Return BestRouteAssignment
    End
EndProcedure
```

Pseudocode Sorting Algorithm

```
Procedure FastBiteSortingBasedAssignment()

  Procedure FastBiteSortingBasedAssignment_Enhanced()

   Input:
      Orders[]       // Each order has: location, delivery time window
      Riders[]       // Each rider has: ID, capacity, current location
      Graph G(V, E)      // V: locations, E: edges with travel times/distances

   Output:
      AssignedRoutes[]    // Routes assigned to each rider
      TotalDeliveryCost    // Total estimated delivery cost (like GA fitness)

   Begin
      // Step 1: Sort orders by earliest delivery time window
      SortedOrders ← Sort(Orders, by = EarliestDeliveryTime)

      TotalCost ← 0

      For each order in SortedOrders Do
         BestRider ← NULL
         MinCost ← ∞
```

```
For each rider in Riders Do
If rider.capacity_remaining > 0 Then
cost ← EstimateTravelCost(rider.current_location,
order.location, Graph)
If cost < MinCost Then
MinCost ← cost
BestRider ← rider
EndIf
EndIf
EndFor

If BestRider ≠ NULL Then
Assign order to BestRider
Update BestRider route and capacity
TotalCost ← TotalCost + MinCost
Else
TotalCost ← TotalCost + PenaltyCost // Unassigned order penalty
EndIf
EndFor

Return AssignedRoutes, TotalCost
End
EndProcedure
```

ALGORITHM ANALYSIS

Algorithm Correctness

Determines if the algorithm produces valid delivery routes

Must satisfy:

- Rider capacity limits
- Delivery time windows
- No duplicate assignments

COMPARISON BETWEEN BOTH ALGORITHM

Graph + Genetic Algorithm (GA) Correctness

Uses a fitness function that penalizes:

- Overcapacity riders
- Missed delivery windows
- Duplicate assignments

Evolves toward better solutions using:

- Selection
- Crossover
- Mutation

Encourages diverse solutions by avoids local optimum

Produces highly valid and optimized routes

COMPARISON BETWEEN BOTH ALGORITHM

Sorting-Based Algorithm Correctness

Greedy approach: Assigns based on earliest time + lowest cost

Based on:

- Cost tracking
- Penalty for unassigned orders

Does not check for global constraint satisfaction

May lead to:

- Unassigned orders
- Overloaded riders
- Suboptimal distribution

COMPLEXITY ANALYSIS

Graph + Genetic Algorithm (GA) Complexity

General Time Complexity

$O(G \times P \times E)$

- G = Max generations
- P = Population size
- E = Evaluation time per solution (based on delivery points & riders)

Best Case

- $O(G \times P)$
- Fast convergence, minimal constraint checking

Average Case

- $O(G \times P \times n)$
- Fitness computed across n delivery points

Worst Case

- $O(G \times P \times n^2)$
- Multiple overlapping constraints and large solution space

Works well with heuristic search to avoid brute-force

Sorting-Based Algorithm Complexity

Main operations:

- Sorting orders: O(n log n)
- Assigning orders: O(n × r) where (n = orders, r = riders)

Best Case

- O(n log n + n)
- Orders fit perfectly, no penalties

Average Case

- O(n log n + n × r)
- Moderate capacity & delivery time variations

Worst Case

- O(n log n + n × r + penalties)
- Orders unassigned due to full capacity

Faster but lacks adaptability or optimization

Example Comparison

At same time, we have

- **12 orders**
- **3 riders**

Graph + Genetic Algorithm

- assign the order evenly to riders [4, 4, 4]. no penalties

Sorting-based Algorithm

- may assign the order unevenly [6, 4, 2]. causes penalties

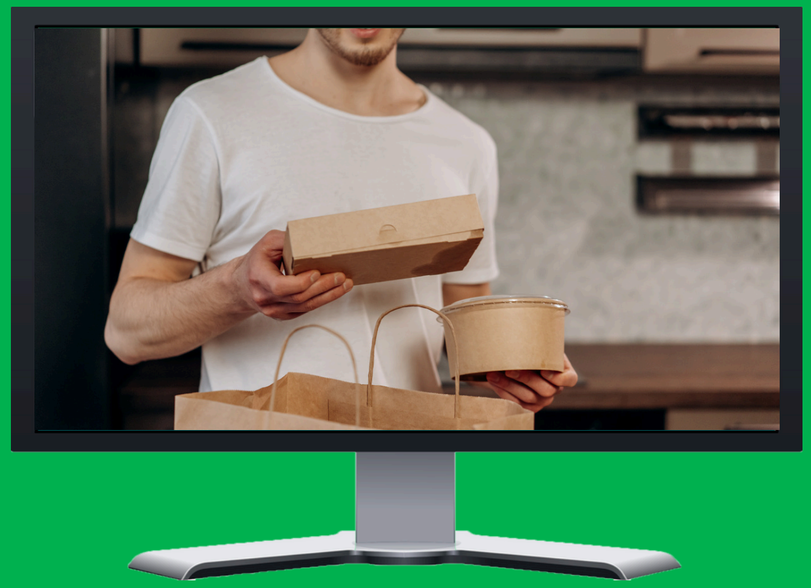
✅ GA: More flexible, scalable, handles constraints better

⚠️ Sorting: Simpler and faster, but less adaptive and not optimal

⟶ For complex delivery scenarios. **GA** is preferred

Testing and Result

PURPOSE OF TESTING



i) Evaluate Graph + Genetic Algorithm (GA) and Sorting-Based Assignment (SBA).

ii) Verify:

- All customer orders are assigned
- Rider capacity limits are respected
- Delivery cost (fitness) is minimized

iii) Compare efficiency and correctness of both algorithms

SCENARIO

- 1 Rider Hub (ID 0)
- 8 Orders → Customer ID 1 to 8
- 3 Riders (capacity = 4 orders each)

RESULT - GRAPH + GENETIC ALGORITHM

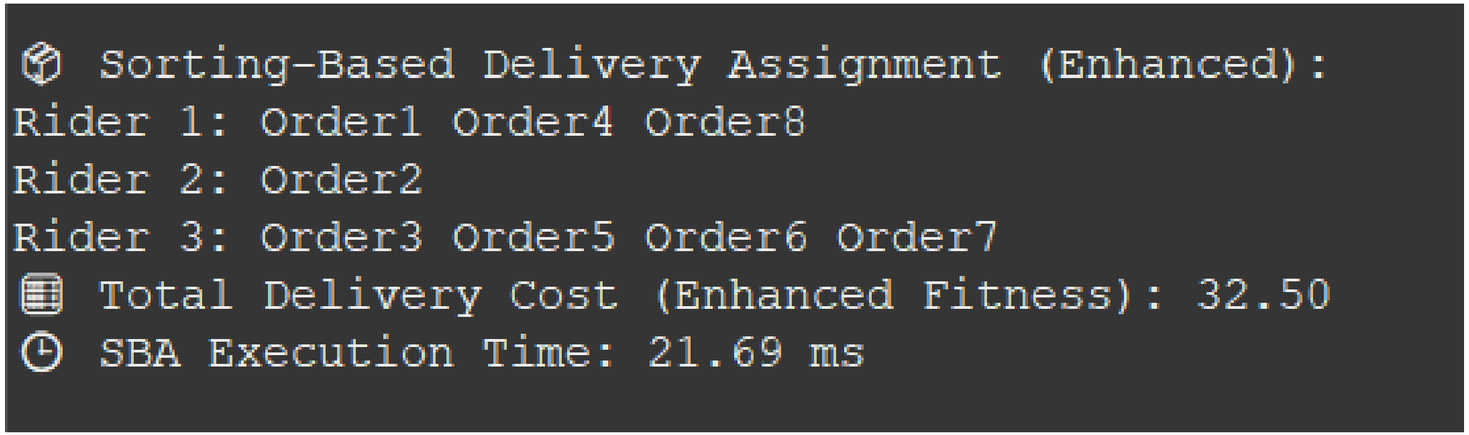
- All 8 orders assigned without conflict
- Rider capacity respected
- Total Cost: 8.00 which shows a very efficient solution.

```
📦 Genetic Algorithm - Best Multi-Rider Delivery Plan:
Rider 1: [4, 5]
Rider 2: [6, 7]
Rider 3: [8, 3, 2, 1]
🧮 Total Delivery Cost (Fitness): 8.00
🕒 GA Execution Time: 57.96 ms
```

Sample Output for GA

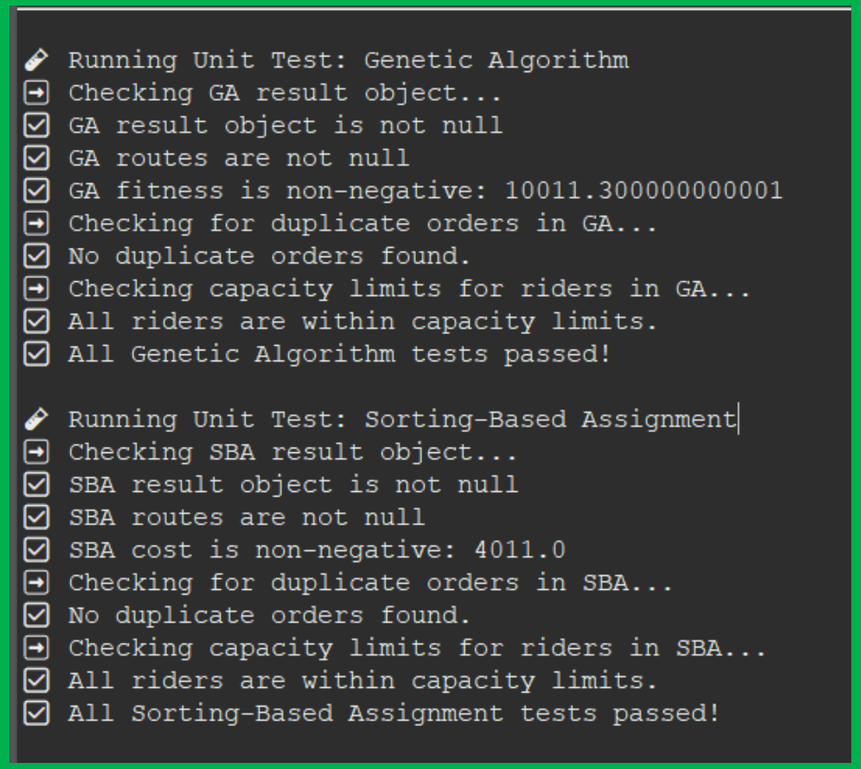
RESULT - SORTING-BASED ASSIGNMENT (SBA)

- Orders sorted by time, assigned to closest available rider
- Uneven load distribution
- Total Cost: 32.50 because SBA is greedy, it makes quick local decisions but doesn’t optimize the overall route.



Sample Output for SBA

OBSERVATION AND VALIDATION



UnitTesting.java

Observations:

- GA is scalable and optimal in terms of cost and load balancing.
- SBA is functional in term of time but not cost-effective
- SBA can still be used in low-complexity scenarios

Unit Tests Validated:

- Order duplication check
- Capacity constraint check
- Fitness cost accuracy

CONCLUSION

This project successfully addressed the challenge of optimizing food delivery routes using algorithmic approaches. By implementing and testing both a Graph + Genetic Algorithm and a Sorting-Based Assignment method, we were able to evaluate their effectiveness in assigning orders, respecting rider constraints, and minimizing delivery cost. The results show that the Genetic Algorithm offers more optimized and balanced solutions, while the Sorting-Based Assignment provides a simpler and faster alternative. Overall, the project demonstrates that different algorithms can be suited to different delivery needs, depending on whether the priority is efficiency or speed.